Offline Handwritten Signature Forgery Detection using Deep Learning Methods

Final Year Project Final Report

A 4th Year Student Name

Phạm Sơn Bách

Nguyễn Huy Đức

Instructor

Dr. Phan Duy Hùng

Bachelor of Computer Science

Hoa Lac campus - FPT University

30 Aug 2021

.

Acknowledgement

We would like to thank our instructor, Dr. Phan Duy Hùng for his patience and time, and for instructing and advising us enthusiastically.

We would like to thank all at my University, FPT, for giving us the best environment to study and grow over the years.

We would like to thank our friends in CS1302 and CS1301, for letting us meet amazing people and learn a lot from them.

We always remember our family's encouragement and support. Thanks to them, we have the will, the energy and the confidence to pursue our goals.

Abstract

Offline signature verification is one of the most challenging tasks in biometric authentication. Despite recent advances in this field using image recognition and deep learning, there are many remaining things to be explored. The most recent technique, which is Siamese Convolutional Neural Network, has been used a lot in this field and has achieved great results. In this thesis, we develop an architecture that combines the power of Siamese Triplet CNN and a stack Fully connected neural network for binary classification to automatically verify genuine and forgery signatures even if the forged signature is highly skilled. In the challenging public dataset for signature verification BHSig260, our model can achieve a low **FAR = 13.66**, which is slightly better than the SigNet model. Once the final model is trained, the one-shot learning should make it possible to determine if the input image is genuine or fraudulent just from one base image. Therefore, our model is expected to be extremely suitable for practical problems, such as banking systems or mobile authentication applications..., in which the amount of data for each identity is limited in quantity and variety.

Keywords: one-shot learning, Offline signature verification, Siamese Convolutional Neural Network, triplet loss.

Table of Contents

1. Introduction

1.1. Signature verification overview

Biometrics technology is widely used in recent security systems. The aim for this system is to identify a person based on their physiological traits, such as fingerprint, face, iris, etc, or behavioral traits like handwritten signature and voice. Those traits are unique among individuals and very hard to fake or steal. Therefore, Most security systems nowadays always apply biometrics technology to enhance the security.

Biometrics system is mainly used in two scenarios: verification and identification. In the first case, the user of the system claims his or her identity, then provides the biometric sample. The role of the verification system is to check if the user is indeed who he or she claims to be. On the other hand, the user provides the biometric sample for the identification system to identify it among all other users enrolled in the system.

In recent years, modern methods for identity authentication systems have become increasingly popular among people. However, the use of documents is still familiar with many companies and businesses. Therefore, the need for an accurate handwritten signature verification system of businesses is always a top concern.

A signature is a handwritten depiction of a person's name, nick name or symbol. The traditional function of signature is to permanently affix to a document, which plays a role as a physical evidence of the author’s personal witness and certification of the content. One of the reasons for its widespread use is that it is simple, fast, non-invasive and people are familiar with it in daily life.

Signature verification system aims to automatically discriminate if a biometrics sample is indeed of a claimed individual. In other words, it is used to classify if a signature is genius or fraud.

The fraud signatures, which are called forgeries, are commonly classified in three types:

- ***Random (blind) forgery:*** In this case, the forger has no information about the user’s name or signature, and uses his or her own signature instead. It is the easiest to recognize by the clear difference between the overall shape of the forger's and the original's signature.

- ***Simple forgery:*** In the simple case, the forger knows only the user's name, and creates the forgery signature of the user's name with his or her own style. This second case has more similarities to the genuine one, particularly if the users sign with their full name.
- ***Skilled forgery:*** This case is the hardest to recognize because the forgeries are made with full knowledge about the user's name and signature shape. Most effective verification system should detect this case.

Figure 1. *Example of different kinds of forgery from CEDAR dataset*

Based on the signature acquisition method, most recent signature verification systems are categorized in two types: online (dynamic) signature verification and offline (static) signature verification. In the online method, the user's signature is acquired by using an acquisition device like a digitizing table. The online signatures are collected as a sequence over time and contain numerous information, such as pen position, pen inclination, pressure, etc. On the other hand, signatures acquired from the offline method are the digital images of the user's signatures signed in the document after the writing process has completed.

The both above mentioned methods are widely used in many studies. When the online method can contribute more diverse features for the verified process, it requires significant devices and techniques to acquire data, which makes the system more expensive and cumbersome. On the other hand, the data from offline methods are easier to obtain, and the image digital signatures are more suitable for practical problems. However, the lack of informative features compared to the other method and the quality instability of image data make the verification process more challenging.

1.2. Problem and Challenging

A. Problem statement:

The problem of offline signature verification is commonly modeled as follows: given a set of genuine signatures of users, a model is trained to extract meaningful features from them. Then, the model is used for verification: a user claims their identity and provides one or some new signatures, which will be used by the model to classify those signatures as genuine (belong to the claimed individual) or forgery (created by someone else).

There are two main approaches for this task in literature, which are Writer-Dependent and Writer-Independent:

- In a Writer-Dependent approach, each user contributes a set of their genuine signatures as the training data (and, often, some genuine signatures from others or the skilled forgery signature as negative samples) to extract some identical features from those signatures for training a binary classification. This approach is very straightforward and usually performs very well on verifying the signature of the user who has their signature trained through the system. However, real-world signature verification, the process of setting new user enrollment is very frequent; and for each new user enrollment, the model must be trained again to learn the features of them. That leads the system to become more and more cumbersome and infeasible to maintain, especially when there may be millions of users in the system.

- On the other hand, Writer-Independent try to learn an efficient representation of signatures from a set of numerous genuine and forged signatures from a huge amount of people. This method is expected to extract distant features between genuine and fraud signatures in general, regardless of signer's factor. Therefore, it is highly flexible, scalable and practical compared to the previous approaches. However, with the variety of linguistic features and style of signatures, it is so rich and challenging to create a universal discriminative representation of signatures to achieve this goal, and no particular feature extraction method has been found to solve this problem, according to [13].

B. Challenges:

In signature verification, one of the most important and familiar challenges for this task is that handwritten signatures have high intra-class variability, which means there are always differences between each handwritten signature from the same person. Compared to other biometric traits like fingerprint, iris or face, handwritten signatures show a large variability between samples, which can be illustrated in Figure 2. This issue leads to an aggravation when considering skilled forgeries, especially with the presence of low inter-class variability in the dataset. In short, when there are large differences between each signature sample (high intra-class

variability) and the forgėies are very skillful that they are nearly the same with the genuine (low inter-class variability), the possibility of confusing the forgeries with the genuine is higher.

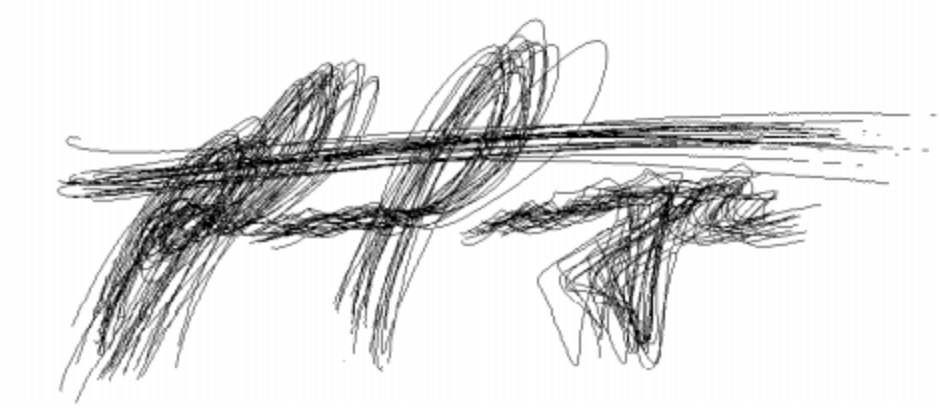

Figure 2. *Superimposed examples of multiple signatures of the same user, which present a high intra-class variability of the signatures of the user* [3].

The second challenge when training an automatic signature verification system comes from the presence of particle knowledge during training. In the realistic scenario, we have only to access genuine signatures for users enrolled in the system during the training phase, according to [3]. During operations, however, we want the system to not only be able to accept the genuine, but also to reject forgeries. To solve that, the forgery signatures for each user should be required for a better classification. But in general, it is not reasonable to require users to provide signatures that forge their own. Even if they can be collected and created from the service provider or from a third party, it would be a challenge to create a good forgeries data which is good enough to be absolutely distinguishable from the genuine signatures.

Thirdly, the amount of training data is always being considered. During the enrollment process in the real application, users are often required to supply only a few samples of their signatures. In the meantime, some approaches like the Writer-Independent approach need a large enough amount of data to perform well. That's why it is hard to make an effective system with the very limited data collected from enrolled users in a real case scenario. Even if there is a large number of users supplying their signatures to the system during the training phase, the performance of the classifier needs to be very good for the new user, for whom supply only few samples of signatures.

Last but not least, like the face recognition system, an effective real-world signature verification system should be able to deal with the one-shot learning problem. Which means the verification application should instantly classify the genuine and fraudulent signatures of a user from just one genuine signature of that individual. However, deep learning algorithms do not work well with a very small dataset, especially if you only have one training example.

1.3. Idea and Motivation

In general, signature verification systems have two phases: feature extraction and classification. In the first phase, the feature obtained from the extraction must be meaningful enough to represent the distinctive features of each individual (Writer-Depent approaches) or show the clear difference between the genuine and forged signatures (Writer-Independent approaches). Then, based on those features, a classifier will be used to make the final prediction.

Since our task focuses mainly on off-line signature, the feature extractor based on a deep CNN (Convolutional Neural Network) may perform better in image data, which is shown through the survey of Hafemann et al. [13].

To overcome the first challenge about high intra-class and low inter-class variability of handwritten signature, a method to transform the original signature's representation into a new one in which the difference between the genuine and the forgeries becomes more obvious can make a big impact on the system performance. In more detail, if this transformation can make the representative factor between the genuine and fraudulent signatures become more clear that the dissimilar representative factor such as feature distance between same genuine signatures is smaller when this one between the genuine and the forged signatures is larger, this model may perform well with one-shot learning problem. This idea has been shown in a recent effective approach for face recognition, FaceNet [9], which will be described later.

One more thing to point out is that in most authentication systems, the ability to accurately detect fraudulent identities is more important than the ability to identify a visitor's identity . For example, in an authentication system that uses fingerprints, it is sometimes acceptable that the system may not recognize the user's fingerprint on the first few attempts, as long as it is able to recognize and prevent complete access of any fake fingerprints. However, if there is a possibility that it could misinterpret a fake fingerprint as the real one and grant access to that forged fingerprint, it would be a really big deal and this system would have to upgrade its security capabilities.

1.4. Related work

In 2008, Impedovo et al. [1] provided a state of art in automatic signature verification, with specific attention to most outstanding advancements in machine learning. Then, Shah et al. [2] presented a survey about the critical evaluation of 15 techniques applied on offline signature verification systems, which classify each work according to the feature extraction methods, classifiers and overall strengths and limitations of the systems. However, these reviews do not update to capture more recent trends, in particular the usage of deep learning methods, which have demonstrated superior results in multiple benchmarks.

Comprehensive surveys and state of the art reviews of the recent literature can be found in the works of [3, 4]. According to these reviews, most deep learning methods for this task can archive results with high accuracy on a large dataset. Beside, among numerous researchers, a growing number of researchers

have realized that "learning with small datasets" will be a key factor for the success of signature verification in real practical scenarios. Authors such as Bouamra et al. [5] try to train the model with a small number of signature samples through a one-class support vector machine (OC-SVM) classifier. Hafemann et al.

Type	Features and algorithm	#Refs	FRR	$FAR_{skilled}$	AER	EER
WD [7]	Graph Matching	16	7.7	8.2	7.9	
WI [38]	Morphology (SVM)	1	12.39	11.23	11.81	11.59
WI [39]	Surroundness (NN)	1		8.33	8.33	8.33
WD [5]	Chain code (SVM)	12	9.36	7.84	7.84	
WD [25]	Curvelet transform (OC-SVM)	12	-	-	5.6	-
WD [28]	Feature learning (SVM)	12	-	-	-	4.63

[6] propose a solution based on a meta-learning approach.

Table 1. *State-of-the-art performance on CEDAR dataset in* [3].

In recent years, many interesting techniques do not rely on hand-engineered feature extraction anymore. With the rise of many deep learning approaches, the ability to execute more meaningful feature representation from raw data (like pixels, in case of images) by itself can promote faster and efficient learning, which helps researchers to save significant amounts of work. In this field, Khalajzadeh et al. [7] applied CNNs for Persian signature verification, but only considered random forgeries in their tests.

Koch et al. [8] firstly propose an approach about Siamese network and one-shot learning for image recognition. This approach outperformed many available methods by previous authors. Moreover, they have argued that this approach can extend in other domains, especially for image classification.

Method	Test
Humans	95.5
Hierarchical Bayesian Program Learning	95.2
Affine model	81.8
Hierarchical Deep	65.2
Deep Boltzmann Machine	62.0
Simple Stroke	35.2
1-Nearest Neighbor	21.7
Siamese Neural Net	58.3
Convolutional Siamese Net	92.0

Table 2. *Comparing best one-shot accuracy from each type of network against Convolutional Siamese Net* [8].

Schoff et al. [9] came up with the same idea of applying the Siamese network, but in a different way. In this research, they propose a new system called **FaceNet**, which learns a way to map face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. They achieved high accuracy (**95.12%** ± 0.39) in their published dataset, and also introduced new concepts about harmonic embeddings and harmonic triplet loss to deal with the face classification task.

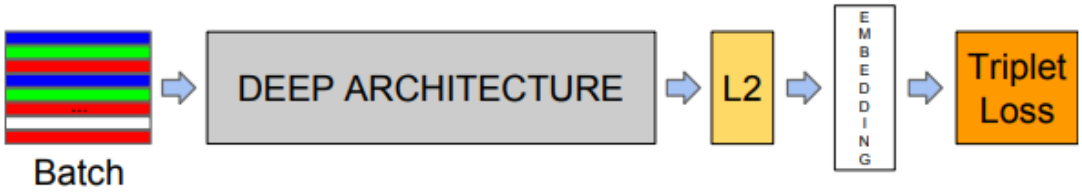

Figure 3. *Model structure of* ***FaceNet***

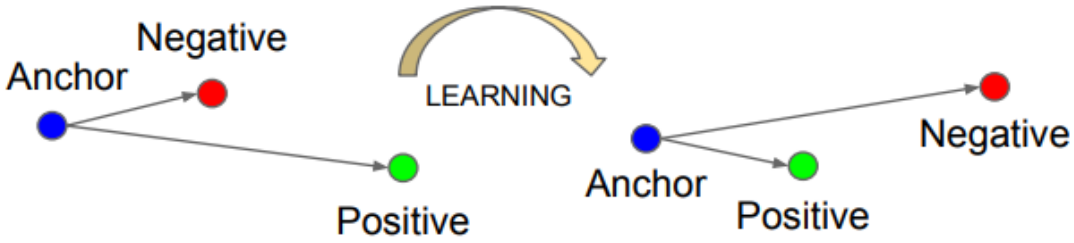

Figure 4. *Intuition of how* ***Triplet loss*** *work on* ***FaceNet***

The above approaches [8, 9] both apply Siamese structure including deep CNNs for getting a new representation of image feature, which will be used to classify tasks by comparing them over their distance between each other in new vector space. Those techniques are very flexible and do not need a dataset with millions of samples to make them work.

Dey et al. [10] applied Convolutional Siamese network with contrastive loss function for writer independent offline signature verification and their system performs very well on cross dataset for this task. In addition, their approach has surpassed the state-of-the-art result on most of the benchmark Signature datasets, which is encouraging this technique for further research. However, when performing evaluation across different datasets, the accuracy of their system gradually decreases on the datasets which have more distinctive features compared to the training dataset.

Databases	State-of-the-art Methods	#Signers	Accuracy	FAR	FRR
CEDAR Signature Database	Word Shape (GSC) (Kalera *et al.* [5])	55	78.50	19.50	22.45
	Zernike moments (Chen and Srihari [22])	55	83.60	16.30	16.60
	Graph matching (Chen and Srihari [12])	55	92.10	8.20	7.70
	Surroundedness features (Kumar *et al.* [8])	55	91.67	8.33	8.33
	Dutta *et al.* [13]	55	**100.00**	**0.00**	**0.00**
	SigNet	55	**100.00**	**0.00**	**0.00**
GPDS 300 Signature Corpus	Ferrer *et al.* [7]	160	86.65	12.60	14.10
	Vargas *et al.* [23]	160	87.67	14.66	10.01
	Solar *et al.* [24]	160	84.70	14.20	16.40
	Kumar *et al.* [8]	300	86.24	13.76	13.76
	Dutta *et al.* [13]	300	**88.79**	**11.21**	**11.21**
	SigNet	300	76.83	23.17	23.17
	SigNet (unskilled forged)	300	65.36	34.64	34.64
GPDS Synthetic Signature Corpus	Dutta *et al.* [13]	4000	73.67	28.34	27.62
	SigNet	4000	**77.76**	**22.24**	**22.24**
Bengali	Pal *et al.* [11]	100	66.18	33.82	33.82
	Dutta *et al.* [13]	100	84.90	15.78	14.43
	SigNet	100	**86.11**	**13.89**	**13.89**
Hindi	Pal *et al.* [11]	100	75.53	24.47	24.47
	Dutta *et al.* [13]	100	**85.90**	**13.10**	**15.09**
	SigNet	100	84.64	15.36	15.36

Table 3. *Comparison of SigNet with the state-of-the-art methods on various signature databases in* [10].

Chhabra et al. [11] have constructed an interesting method based on the Convolutional Siamese Net architecture from **FaceNet** [9] and triplet loss concept to solve the one-shot problem for off-line signature verification. Their model has high accuracy and generalizability on the public signature database. However, they do not show the evaluation result in various databases and the performance comparison with other state-of-the-art methods.

1.5 Contribution

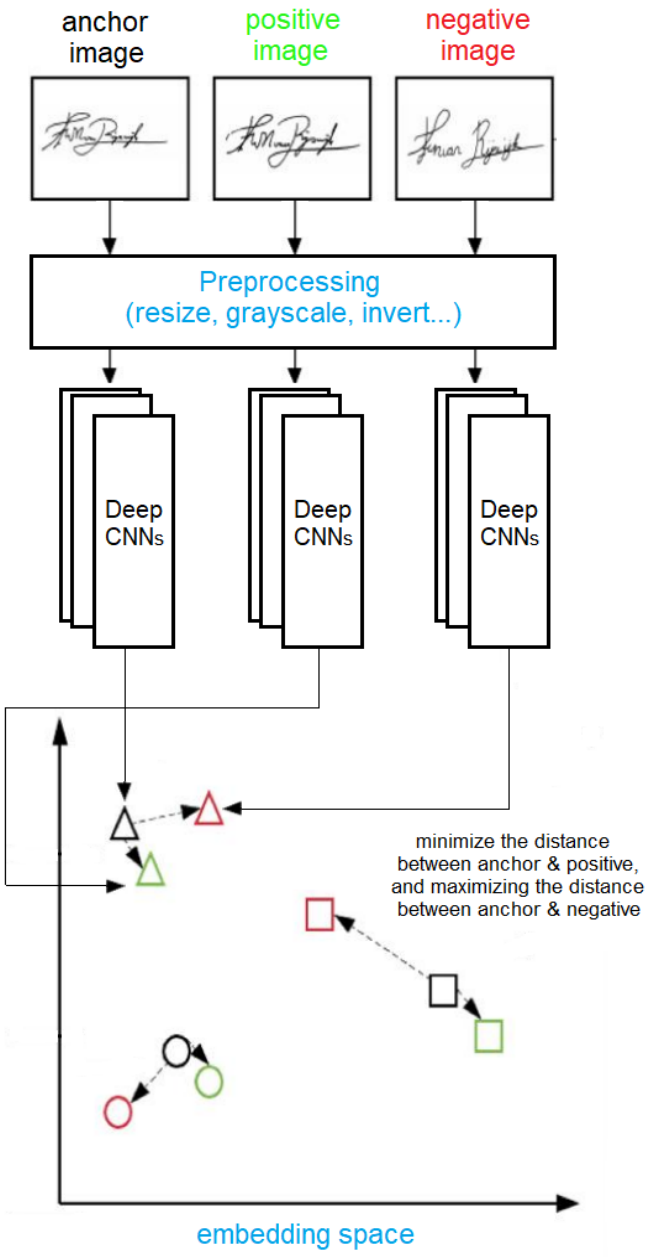

Figure 5. *Deep Triplet Ranking CNN Network* [11]

Inspired by the work of Chhabra et al. [11], we use the similar network architecture of Deep Triplet Ranking CNN Network, which is illustrated in ***Figure 5***, and add some modifications to try to increase the robustness and the performance.

In our work, we try to improve the above architecture by applying more robust transfer learning models and more efficient classification methods. We also evaluate it on more different datasets to see if it is actually generalizative.

Please note that our work focuses mainly on forgery detection in signature verification systems with one-shot learning strategy, which means our model may not perform well on recognition identity from signature but to eliminate forgeries as much as possible.

The rest of this thesis is organized as follows: In Section 2, we will give a detailed overview of the public datasets used and present our methods for preprocessing. Section 3 describes our network architecture and some relative methods. Section 4 is where our results on the evaluation process will be shown and compared with other approaches. Finally in Section 5, we conclude our result and define the future work.

2. Dataset and Preprocessing

2.1. Dataset overview

For comparison purposes, we decided to use some well-known public dataset from many other studies in this task. There are 3 public datasets used in our thesis: CEDAR signature dataset, SigComp 2009-2012 signature dataset and BHSig260 signature dataset. The reason we choose those dataset is that they are public, free to access, and contain skilled forgeries.

2.1.1. CEDAR

CEDAR Signature is a dataset of off-line handwritten signatures. In this, each of 55 individuals contributed 24 signatures thereby creating 1,320 genuine signatures. Some were asked to forge three other writers' signatures, eight times per subject, thus creating 1,320 forgeries. Each signature was scanned at 300 dpi gray-scale and binarized using a gray-scale histogram. The database has 24 genuines and 24 forgeries available for each writer.

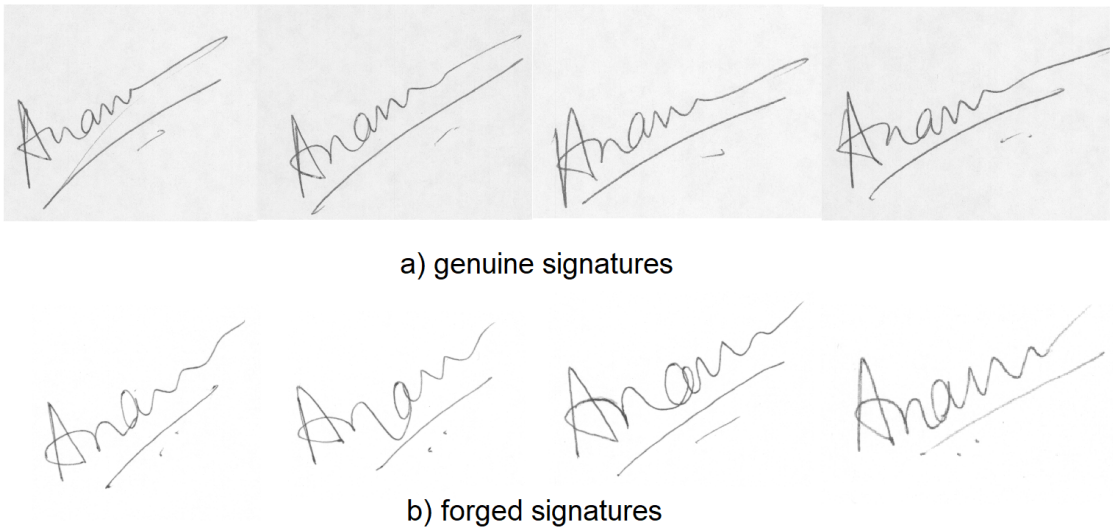

Figure 6. *Examples of CEDAR signatures.*

In this dataset, salt pepper noise has been added and the signatures are skewed in different directions.

2.1.2. BHSig260

BHSig260 dataset contains the offline signatures of 260 persons, in which 100 persons were signed in Bengali and the remaining were signed in Hindi. For each of the signers, 24 genuine and 30 forged signatures are available. This results in 2400 genuine signatures and 3000 forged signatures in Bengali, and 3840 genuine and 4800 forged signatures in Hindi. Our experiment is only in the Bengali signatures datas from this dataset.

Even though these signatures are already in binary (black and white) form, they still contain salt pepper noise with sparse density compared to CEDAR.

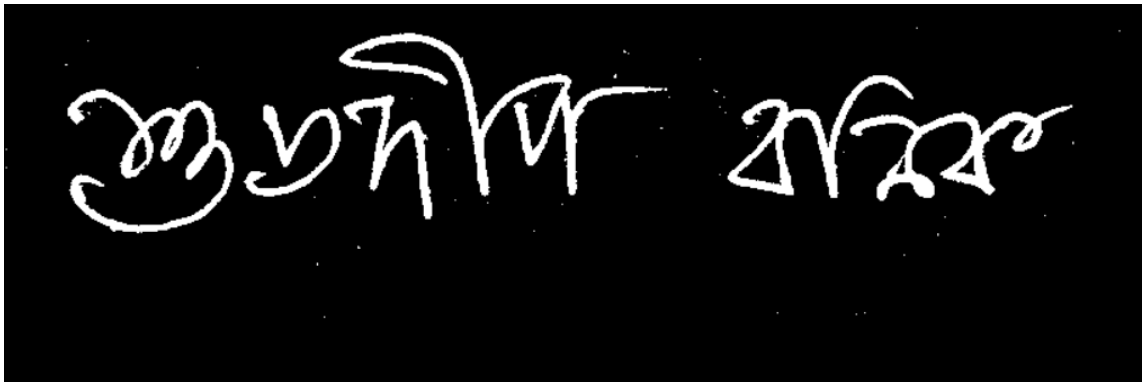

Figure 7. *An example of signature in BHSig260 with inverted index value. We can see that there are some white small dots (salt noise) in the background of the image.*

2.1.3 SigComp2013

SigComp is the public dataset used in the ICDAR Signature Verification Competition. This competition always updates the dataset every year the competition is held. Therefore, there are some versions of the available dataset from this competition in various languages and styles. Here, we use SigComp2013, which is a combination of SigComp2009 and SigComp2011, for the training phase and evaluate it on the test set from SigComp 2010 [15]. However, in those datasets, we only use the offline Dutch signatures of SigComp2011.

Dataset Name	Number of Users	Genuine Signature for each user	Forgeries for each user
SigComp2009	12	5	150
SigComp2011	54	12	24

Table 4. *SigComp2013 breakdown.*

2.2. Preprocessing

First, we can see that color factor is not important when comparing the dissimilarity of genuine and fraudulent signatures. That is why we convert all signatures to grayscale.

As mentioned earlier, our dataset contains salt pepper noise in most signature images. However, the noise density in each image is quite sparse. By using a Median filter with kernel size 3x3, most salt and pepper noise are easily eliminated.

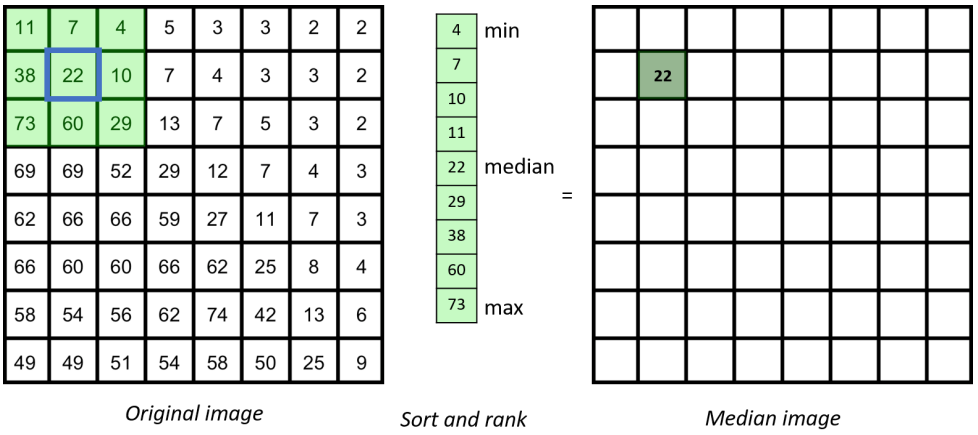

Figure 8. *Median filter illustration.*

After removing noise, we use Otsu threshold method to convert the gray images to be the black and white (only including the pixel value of 0 or 255). In addition, we invert the image to make the background become zero and only the signature's area has pixel value 255.

Finally, we resize all signature images to a fixed size for feeding into the network. With the signature of CEDAR and SigComp, we decide to use the shape 128x128 as the work on [11]. However, most images do not have square size, and some of them have a very long width compared to the height. To ensure that the overall shape of the signature in the image is not deformed after resizing, we add the zero-padding (black padding) to change the shape of the rectangular image into square shape before resizing it into the shape 128x128 like the other dataset.

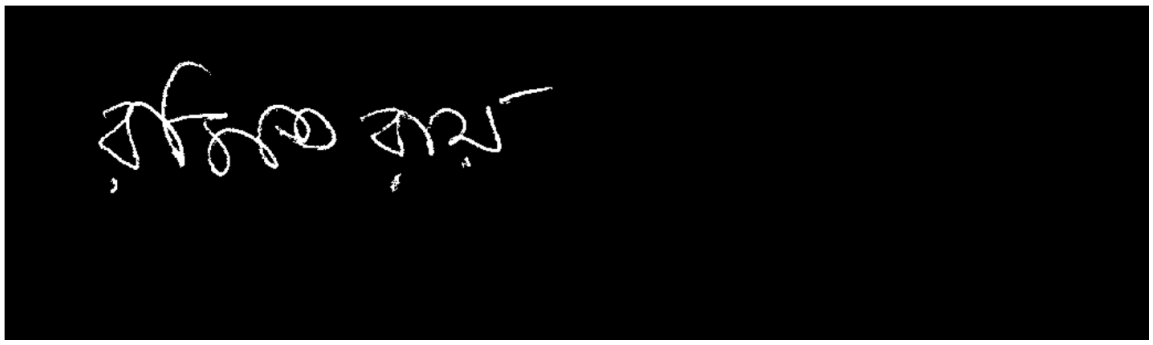

Figure 9. *An example of BHSig260 that has a long width.*

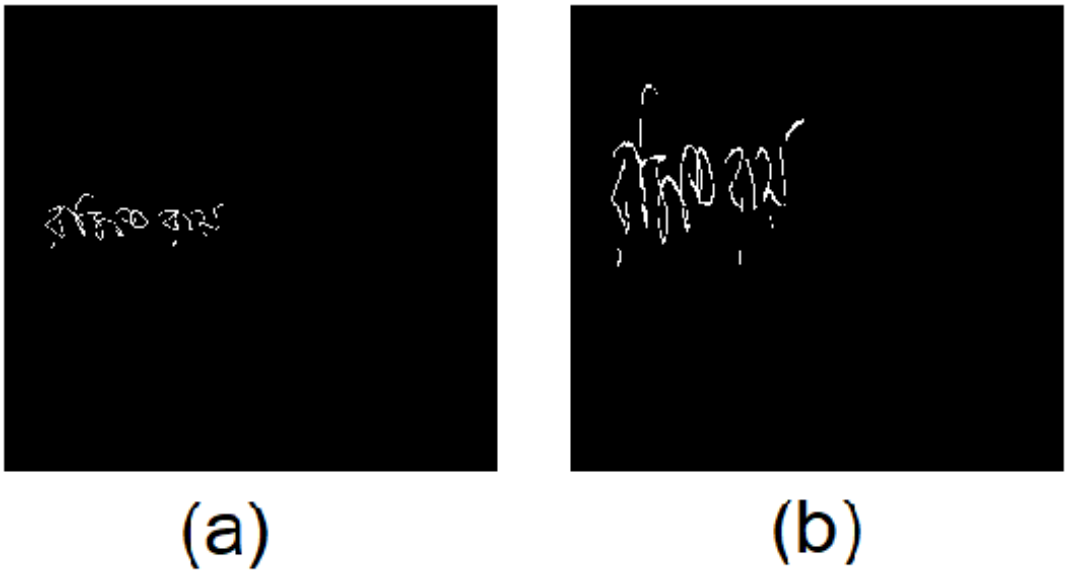

Figure 10. *Resizing image with* **(a)** *and without padding* **(b).**

3. Methodology

3.1. Deep triplet ranking CNN architecture

3.1.1. Siamese Network with image

The Siamese nets were first introduced in the early 1990s by Bromley and LeCun to solve signature verification as an image matching problem [16]. Siamese network is a class of neutral network architectures that contain two or more identical subnetworks. The "identical" here means each subnetwork has the same configuration with the same parameters and weights. Parameter updating is mirrored across all subnetworks.

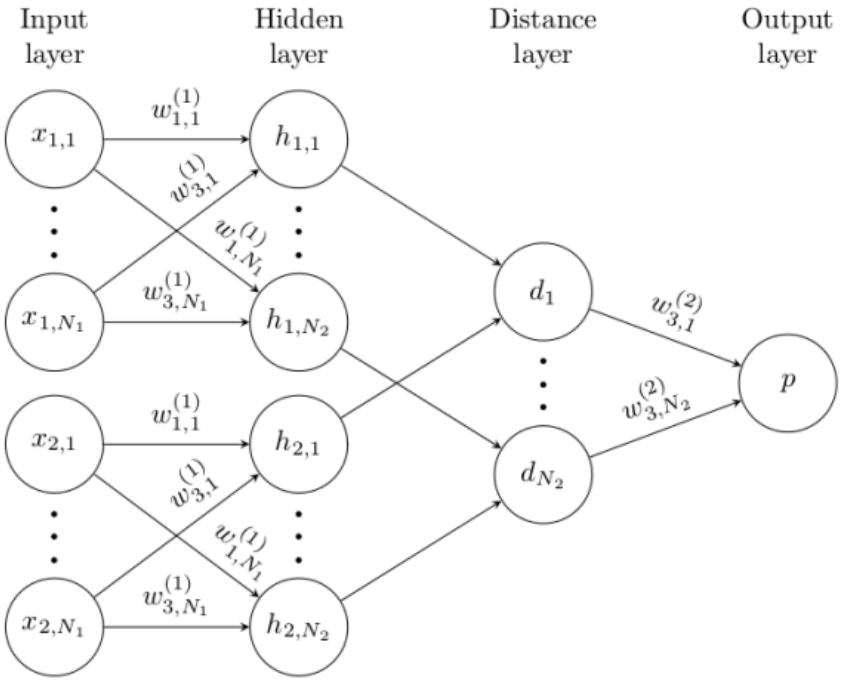

Figure 11. *A simple 2 hidden layer siamese network for binary classification with logistic regression p. The structure of the network is replicated across the top and bottom sections to form twin networks, with shared weight matrices at each layer* [8].

The idea behind this architecture is that with the sysmetric architecture, each subnetwork can compute a higher feature representation of multiple inputs at the same time. Then, those feature representations go through a distance layer to compute the similarity between them and an energy function at the top will make the prediction from that similarity (***Figure 11***). By sharing weight between identical subnetworks, this technique guarantees that two extremely similar input images could not be possibly mapped by their respective network to very different locations in feature space. Meanwhile, the network is symmetric, so that whenever we present two distinct images to the twin networks, the top conjoining layer will compute the same metric as if we present the same two images but to the opposite twins. With these characteristics, the Siamese architecture is very effective for computing the similarity between data with different classes.

From that idea, the combination of the Siamese net structure with an efficient deep neural network for image features extraction, such as CNN, is very suitable for computing the similarity and dissimilarity between images. Computing the 'distance' between different samples in the high level feature representation with this technique does not require a huge amount of data but still performs well.

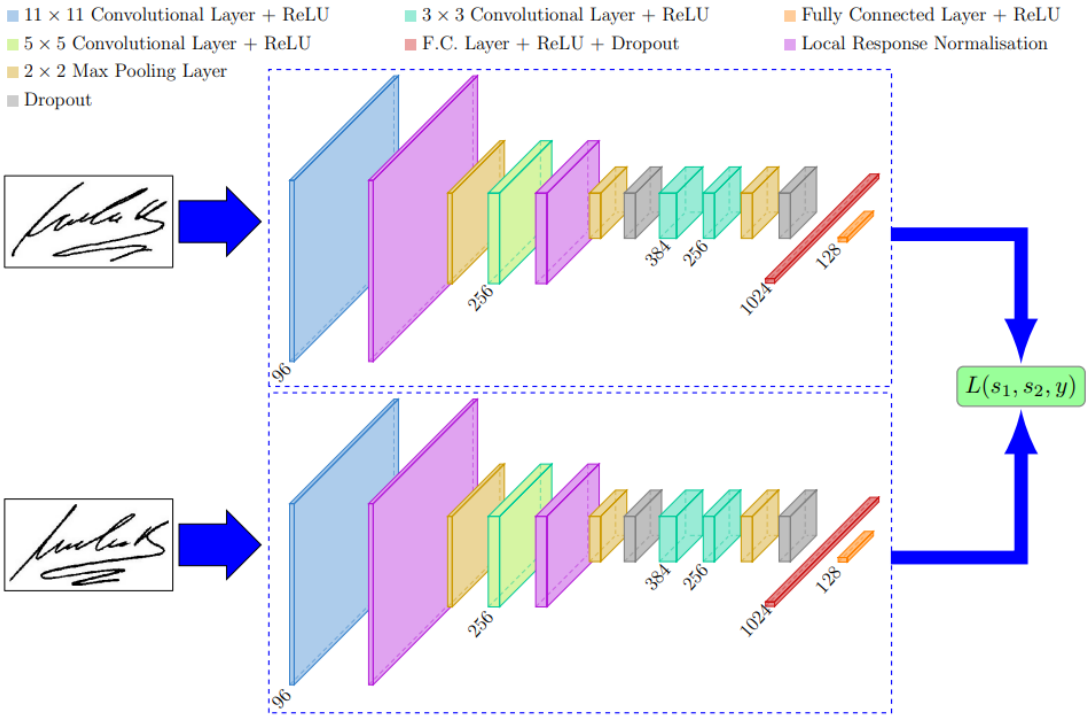

Figure 12. *Example of Siamese CNN model for computing signature similarity in SigNet.* [10]

3.1.2. Architecture breakdown

Like the architecture introduced by Chhabra et al. [11], our system uses 3 sysmetrics deep Convolutional Neural Networks played as the subnetworks in the Siamese architecture. Instead of building a new deep CNNs and training it from scratch, we will use the Xception model [12], of which overviewed architecture is illustrated in ***Figure 13***.

Because the Xception model has been pre-trained in ImageNet, we perform a transfer-learning strategy to reduce time for the training process: first we update the model with the weight trained on ImageNet dataset and remove the fully-connected layers on top, then we freeze all layers except the exit flow for train on our dataset. This strategy helps our model have the ability to extract the high feature representation of the image (from the freezed layers) while still being able to learn the important feature from the signature image in our dataset.

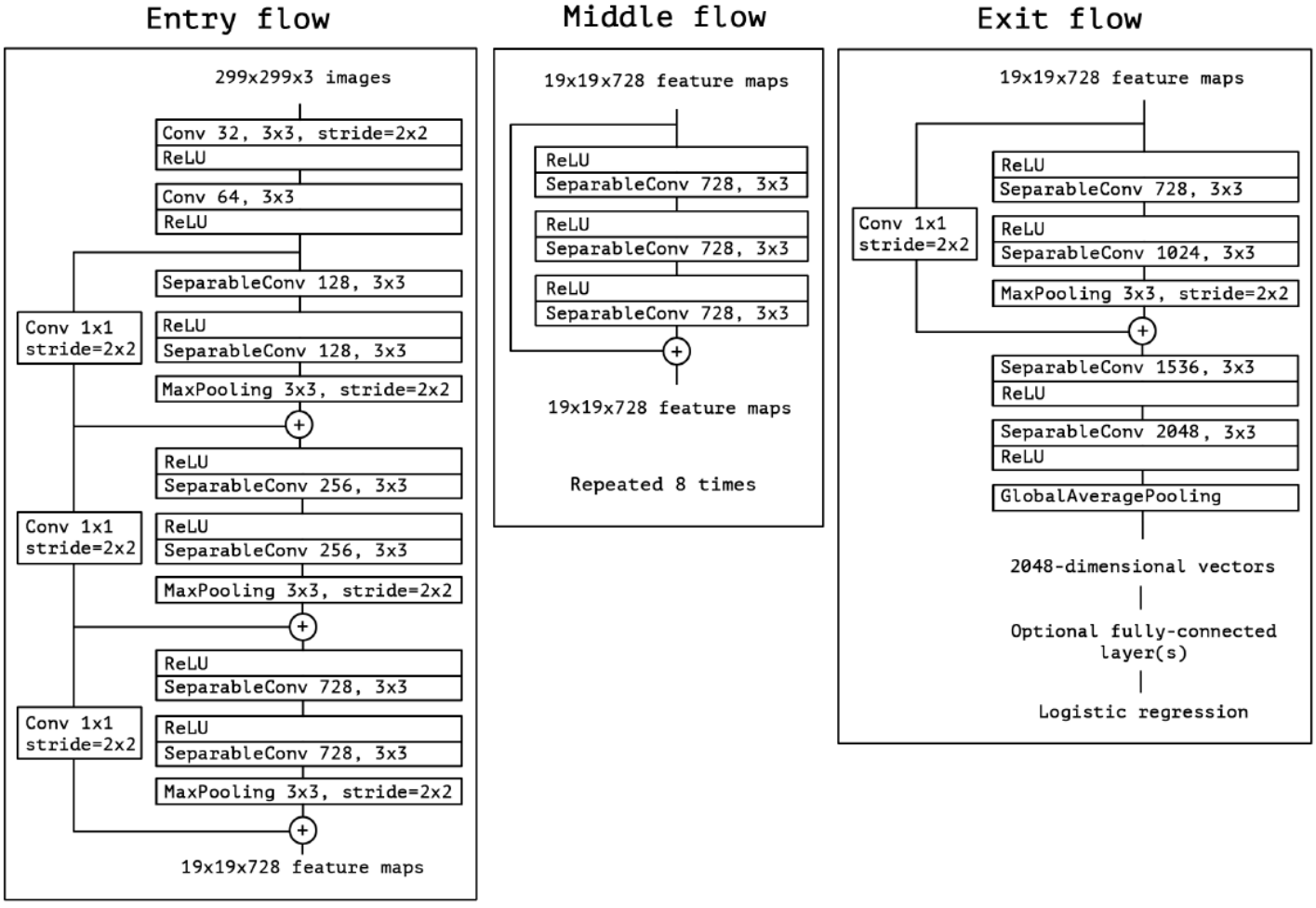

Figure 13. *Xception architecture overview.*

The reason we choose Xception is that it outperformed some most popular recent networks like VGG-16, ResNet and Inception V3 (***Table 5***). Moreover, it has fewer parameters than Inception V3, which can save more time on the training process (***Table 6***).

	Top-1 accuracy	Top-5 accuracy
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	**0.790**	**0.945**

Table 5. *Classification performance comparison of Xception with the other.*

	Parameter count	Steps/second
Inception V3	23,626,728	31
Xception	22,855,952	28

Table 6. *Size and training speed comparison*

Given the architecture details for the extraction model (see ***Figure 14***), our model accepts the input which has a form of three signature images categorized as: anchor (an individual's genuine signature), positive (another genuine signature of the same individual) and negative (a forged signature of the same individual). After the preprocessing, those triplet images are fed through the triple subnetworks in which each is constructed by transfer-learning Xception top up with a stack of BatchNormalization layer and a fully-connected neural network with ReLU activation function.

When the model gets the new feature representation (embedding) of each image in the triplet, a Global max pooling layer is used to reduce the shape of them to 1 x n matrix. After being concatenated, our network uses a Lambda layer to perform both the distance computing process and triplet loss calculating process.

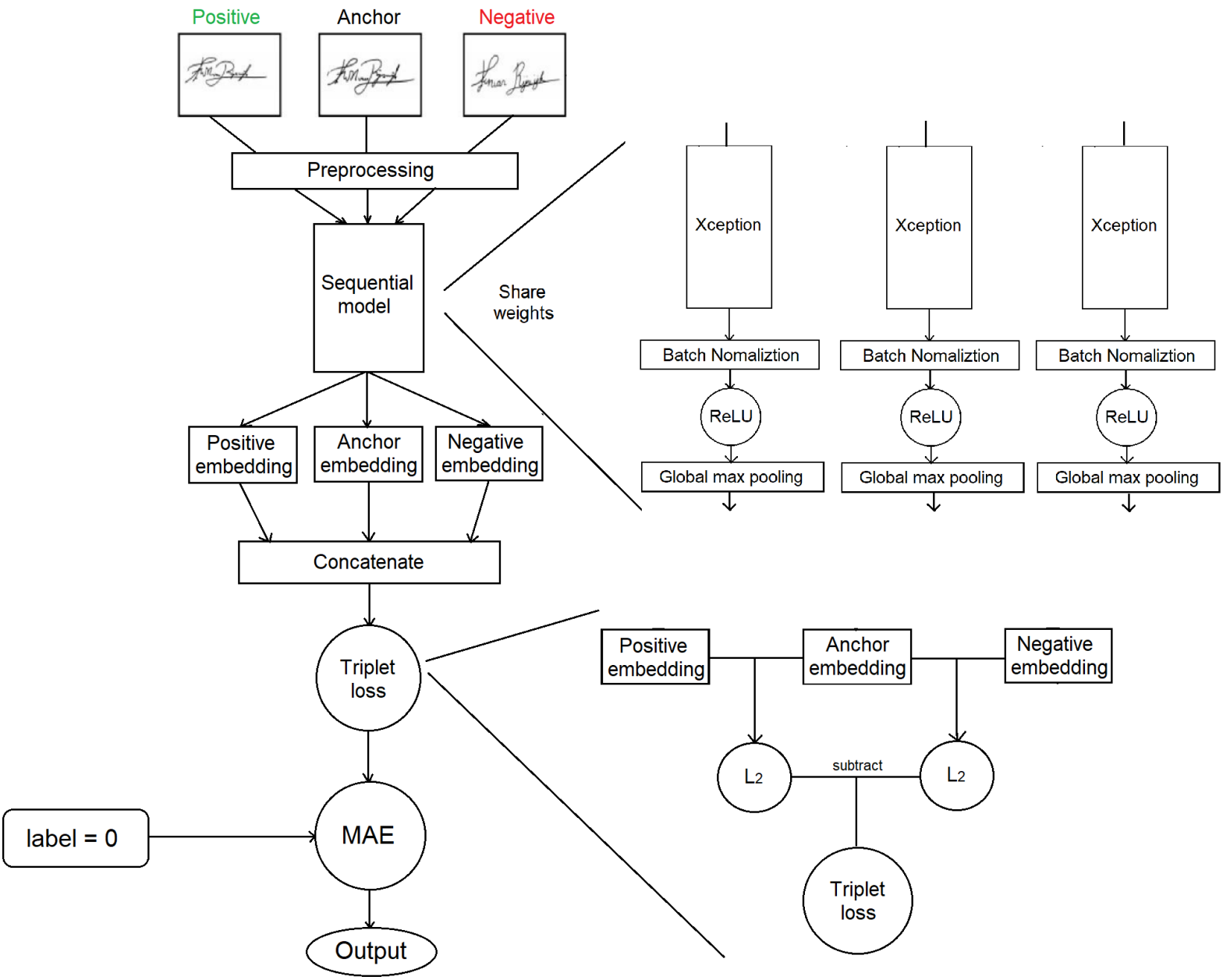

Figure 14. *Our Deep triplet ranking CNN architecture with Xception.*

For the optimization phase, we use Adam Optimizer with the learning rate equal to 1e-5 and mean absolute error for backpropagation to get the final encoding.

Layer (type)	Output Shape	Param #	Connected to
anchor (InputLayer)	[(None, 128, 128, 3)	0	
positive (InputLayer)	[(None, 128, 128, 3)	0	
negative (InputLayer)	[(None, 128, 128, 3)	0	
sequential (Sequential)	(None, 4, 4, 1024)	22967848	anchor[0][0] positive[0][0] negative[0][0]
Global_anchor (GlobalMaxPooling	(None, 1024)	0	sequential[0][0]
Global_positive (GlobalMaxPooli	(None, 1024)	0	sequential[1][0]
Global_negative (GlobalMaxPooli	(None, 1024)	0	sequential[2][0]
concatenate (Concatenate)	(None, 3072)	0	Global_anchor[0][0] Global_positive[0][0] Global_negative[0][0]
lambda (Lambda)	()	0	concatenate[0][0]

Total params: 22,967,848
Trainable params: 6,851,072
Non-trainable params: 16,116,776

Figure 15. *Structure and layer configuration of our triplet CNN architecture*

3.1.3. Triplet loss

The triplet loss using for this architecture is similar to the triplet loss using in FaceNet [10], which is formulated as:

$$L(x^a,\ x^b,\ x^n,\ \alpha) = \frac{1}{N}\sum_{i}^{N} max\left\{D(f(x_i^a),\ f(x_i^p)) - D(f(x_i^a),\ f(x_i^n)) + \alpha,\ 0\right\} \quad (1)$$

where,

$f(x)$ refers to an embedding of the image x

$x^a,\ x^p,\ x^n$ are the anchor image, positive image and negative image, respectively

$D(f(x^a),\ f(x^p))$ is the Euclidean distance between the $f(x^a)$and $f(x^p)$

α is a constant (or margin) used to make sure that the network does not try to optimize towards the case $D(f(x_i^a),\ f(x_i^p)) = D(f(x_i^a),\ f(x_i^n)) = 0$.

3.1.4 Triplet selection (triplet mining) method

In FaceNet, the authors announced that the triplet selection method is crucial for the model to ensure that it can convergence fase. Looking at the triplet loss equation above (1), let us call the L2 distance between the anchor and the positive is positive distance (d(a,p)) and between the anchor and the negative is negative distance (d(a,n)). There are some possibility when forming a triple image input, which can be categorized into:

- **easy triplets:** triplets which have a loss of 0, because:

$$d(a,p) + \text{margin} < d(a,n)$$

- **hard triplets:** triplets where the negative is closer to the anchor than the positive:

$$d(a,n) < d(a,p)$$

- **semi-hard triplets:** triplets where the negative is not closer to the anchor than the positive, but which still have positive loss:

$$d(a,p) < d(a,n) < d(a,p) + \text{margin}$$

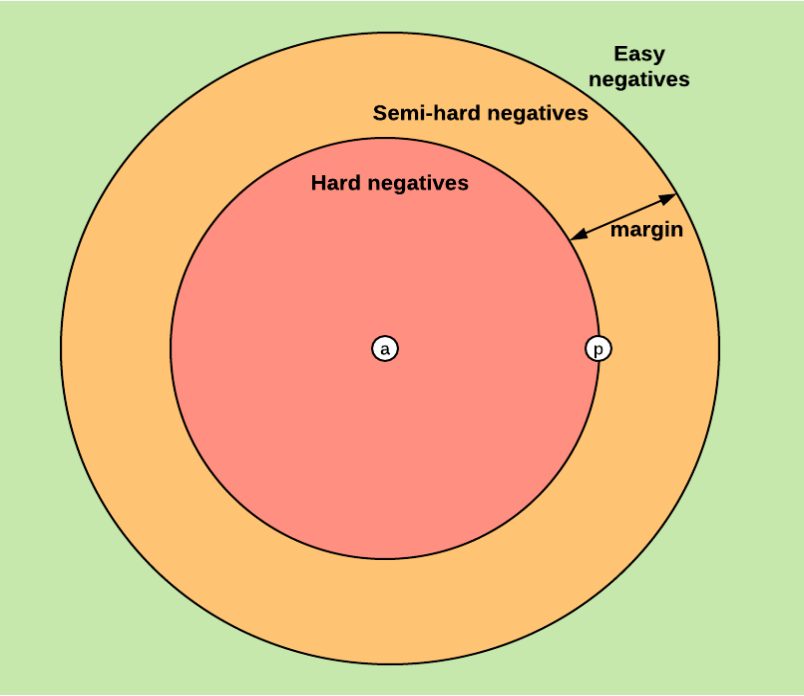

Figure 16. *Regions of embedding space for negatives.*

According to the authors of FaceNet, avoiding selecting easy triplets can help a lot since the model would not learn anything from them. Therefore, with our dataset, we decide to generate all triplets before training instead of doing online mining like the original authors. However, our datasets contain almost skilled forged signatures and are categorized by each user. That's why we select each triplet such that all of them come from the same user to decrease the chance of getting an easy triplet.

3.2. Classification model

From the triplet net, we only use the Sequential part (see ***Figure 14***). The purpose of doing triplet loss in this architecture is to update the weight of the sequential part so that the embedding images have adjusted distance to become closer if they are genuine, or become further away from the forgeries.

Instead of using the embedding to classify, the authors in [11] have the idea to detect forgeries and genuine by calculating the vector difference between those embedding and treat them as the feature set for classification. According to their methods, they first use the Sequential part as an encoder to get the embedding of all genuine and forgeries. After that, they arrange those embeddings in a pairwise manner where each observation is a pair of images, either both of a person's genuine signature, or one of the person's genuine signature, and the other as a forged signature of that person. These will have labels (class) genuine or fraud (0 or 1) assigned to them respectively. They use cross validation to get to the final classifier model taking the corresponding differences between the embeddings of each of the pairs (1024 length difference vector of embeddings) as the feature set and the class labels (genuine/ fraud) as the dependent variable y.

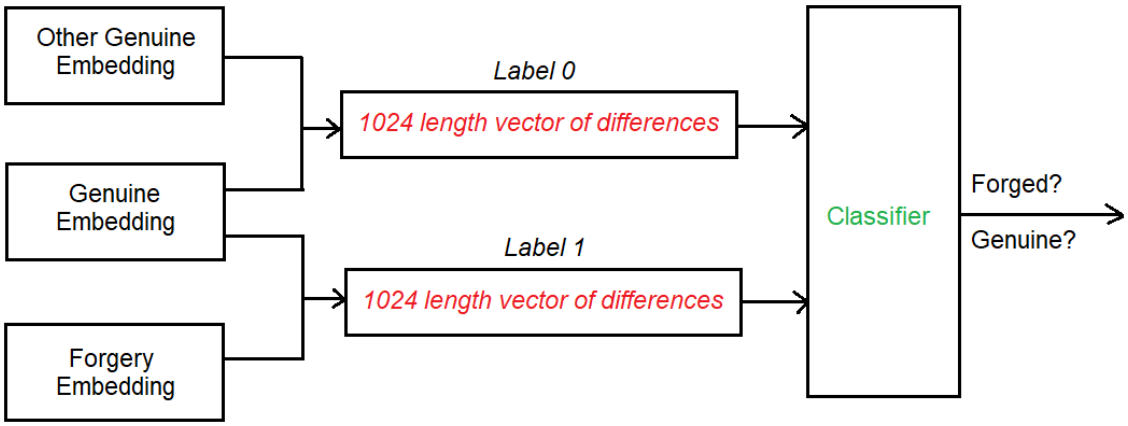

Figure 17. *Classification flow illustration.*

The authors in [11] use logistic regression as the classification for this phase. However, we construct a new binary classification with Fully-connected Layer and ReLU activation function, which is top-up with sigmoid function to calculate the probability that a signature is forged or genuine.

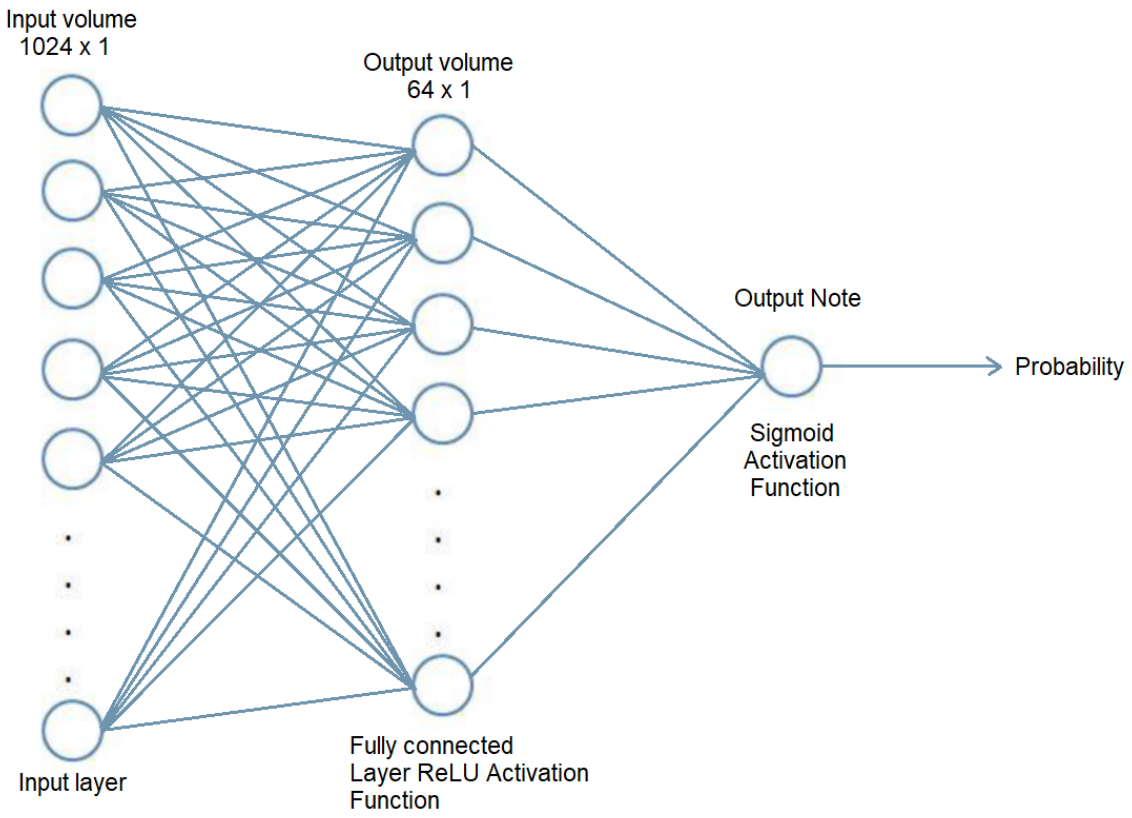

Figure 18. *Our Binary Classification model structure.*

We optimize this classification with **Adam** optimizer and use **binary cross entropy** for computing loss value.

3.3. Final model architecture

After training Deep Triplet CNNs model and the classification, we only use the weight of the Sequential part in the Deep Triplet model (see ***Figure 14***) for constructing the final model, as discussed in [11]. The idea is that the Sequential part in the Deep Triplet model will be an encoder to encode every image that goes through the encoder and return the embedding version of them. After that, we create pairwise with the strategy mentioned above and feed them to the classification.

With the evaluated data, first we need to perform preprocessing for them. Then we create pairs like when we train the classifier and label them. Finally, the new feature from pairs set and label sets go through the classifier to get the final result.

In addition, when this model is used in real applications, they do need more data to train anymore if they have already performed well in training and evaluated phase. The ideal model constructed from our architecture requires only one genuine signature image for each new user. This image should perform exactly the same preprocessing method and feed through the encoder

to get the embedding. Then this embedding will be stored in the database and be marked by an id or any key to query in future. Moreover, we will call them the base signature for each user.

When a new signature image is attempted on the system, if the owner of the new signature claims to be anyone in the system by promoting the id, the system will use that id to track and query the base embedding out for comparison. The system then uses the encoder to get the embedding of the new signature image, and get the vector's difference between the embedding of the base signature and the new signature. Lastly, the classifier will use this vector;s difference as an input to predict the probability if the new image is a forged signature or not. The higher the value of this predicted result, the higher the probability that the new signature is a fake one.

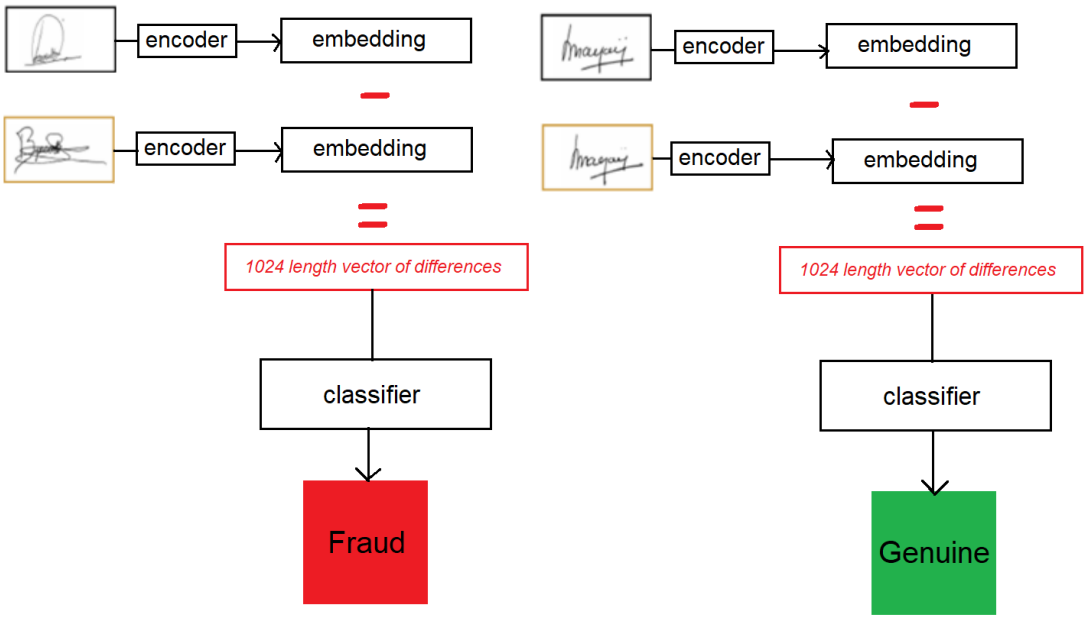

Figure 19. *Testing process of determining whether signature is genuine or fraud*

3.4. Evaluation methods

In the Triplet CNN model, we try to minimize the triplet loss as close to zero as possible. Therefore, our model will achieve its best effect when all the triplet embeddings attained through it (which we call *valid triplet*) can satisfy:

$$d(a,p) < d(a,n)$$

To know if our model can satisfy the above condition, we define a simple accuracy equation for our model as:

$$Accuracy = \frac{number\ of\ valid\ triplets}{total\ number\ of\ triplet} \quad (2)$$

As we mentioned earlier, our model focuses mainly on detecting forgery signatures instead of identifying the user's identity. With that mindset, there is a class that is more important than the other, making the accuracy metric and the normal error rate computation not really suitable for our problem. Instead, we use F-beta Score and Precision - Recall Curve to evaluate our classification model.

To use two above evaluate metrics, we need to form a confusion matrix (***Figure 20***), which is calculated as follow:

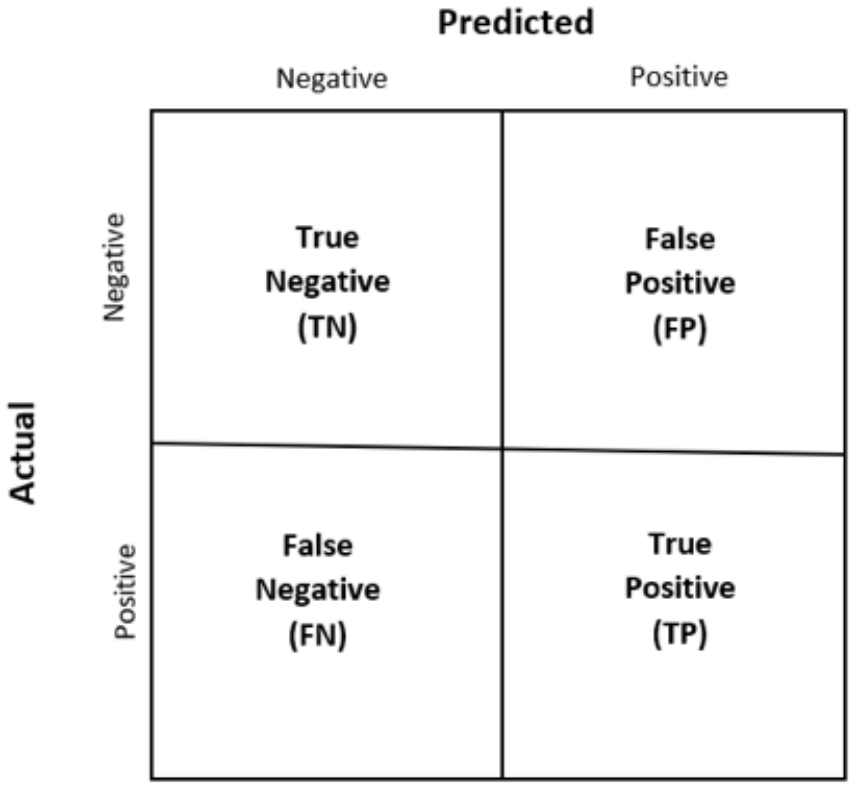

Figure 20. *Illustration of Confusion matrix of sklearn.metrics in python libraries.*

- **F-beta Score:** From confusion matrix (see ***Figure 20***), we can calculate the F-beta Score as:

$$F_{beta} = (1+\beta^2)\frac{precision * recall}{\beta^2 * precision + recall}$$

where,

precision = TP / (TP + FP) recall = TP / (TP + FN)

The more we care about precision, the lower beta should be. Vice versa, the more we care about recall, the higher beta should be.

Because we care more about the precision (also called positive predictive value), which is the ratio between the number of correct positive predictions and the total number of positive predictions, using 0 < beta < 1 may be good. However, we still use Precision-Recall Curve for evaluating and F1 Score for finding the best threshold.

- **Precision - Recall Curve:**

  According to the sklearn library in python, "*The precision-recall curve shows the tradeoff between precision and recall for different thresholds. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall*)". Using this curve, we can see the trace-off flow between precision and recall when we change the threshold many times. From that, we can even decide the best threshold to use instead of the default threshold (0.5). Moreover, we will use the Area Under the Curve (AUC) metric instead of the accuracy metric.

We use the basic evaluation index like False Positive Rate (FPR), which is calculated as the number of incorrect positive predictions divided by the total number of negatives (the same as the definition of FRR). Similarly, the False Negative Rate (FNR) is calculated as the number of incorrect negative predictions divided by the total number of positives (the same as the definition of FAR). In summarize, we have the following formula:

$$FAR = FNR = FN / (FN + TP) = 1 - recall$$

$$FRR = FPR = FP / (FP + TN)$$

Note that in our model, the positive class is the forged signature and the negative class is the genuine signature. In addition, we choose FAR as the most important evaluate model to consider like other studies in this field.

4. Experiment

4.1. Experimental Setup

First of all, we create our dataset by splitting 20% of users and get all signatures of that 20% to form a hold-out test set. That makes our test set disjoint identities with the data from the training set.

When forming the triplet combinations to training the Triplet net, we do the same as the method in [11], which is done by taking an anchor image (genuine signature of a person) and placing it in conjunction with both a positive sample (another genuine signature of the same person) and a negative sample (a forged signature by someone else of the same person). By that strategy, from datasets that have around 648 to 3000 signatures for each class, we can create hundreds of thousands of triplet combinations for training (***Table 7***).

Dataset Name	Number of triplet combinations
SigComp2011 Dutch	33 580
CEDAR	284 832
BHSig260 Bengali	654 120

Table 7. *Number of training triplet combinations among various datasets.*

The training pairs for the classification are formed after we get the embedding of all signatures using the train triplet net. Then we create the pairs, label them (see ***Table 8***) and feed them to the classifier.

With the workflow of the final model (shown in ***Figure 19***.), we have to create the test set different from the train set for classification. Because our model should deal with a one-shot problem, for each user in the dataset, we create a dictionary to store only one of this user's genuine signature embedding (called base embedding) and set their id as the key. After that, we create the pairwise manner by letting each base embed pairing with all genuine and forged signatures of that user and labeling them as 0 and 1, respectively. This pairing method ensures that the test case of our model only lay on the skilled forgeries.

Set(Train/Test)	Pair	Label
Train Users	**Genuine - Genuine**	0
	Genuine - Forgery	1
Test Users	**Base Genuine - Genuine**	0
	Base Genuine - Forgery	1

Table 8. *Pairs labeling method for train and test*

4.2. Results

In work, we train our triplet on each dataset with the same margin = 0.2 on the triplet loss with the learning rate of Adam Optimizer = 1e-5 and using only MAE loss value to validate. The triplet combination is randomly split into a train set and validation set with ratio 2:1, respectively. We use the eq. (2) to calculate the accuracy of our model, which is shown in the ***Table 9***:

Table 9. *Deep Triplet CNN model performance.*

Dataset	Accuracy on training set (%)	Accuracy on validation set (%)	Accuracy on test set (%)
SigComp2011	*95.3*	*95*	*80*
CEDAR	*67.85*	*67.92*	*72.84*
Bengali	*96.46*	*96.36*	*81.8*

The classifier is trained on a labeled feature set, which is constructed as mentioned in the **Section 3.2** and is randomly splitted into a train set and validation set with ratio 2:1 . The ***Table 10*** and ***Table 11*** show the performance of our model on detecting forgery and genuine signatures.

Table 10. *Performance on validation set*

Dataset	AUC (%)	ERR	FAR	PRR
SigComp2011	100	0.00	0.00	0.00
CEDAR	99.86	1.64	2.09	1.15
Bengali	99.99	0.13	0.00	0.00

Table 11. *Performance on test set*

Dataset	AUC (%)	ERR	FAR	FRR
SigComp2011	65.11	50	70.39	18.82
CEDAR	68.21	34.55	52.09	16.15
Bengali	86.16	22.75	18.57	27.97

Beside, we test the performance of our test in the case that the pair Base Genuine - Forgeries is no longer depend on identity anymore, which means we construct the pair from the base genuine signature with all forgery signature in the dataset to get more positive label (the forged signature) for test (***Table 12***)

Table 12. *Performance on test set with random forgeries by other user forged signatures*

Dataset	AUC (%)	ERR	FAR	FRR
SigComp2011	98.41	75	75.75	13.73

CEDAR	96.69	42.84	45	16
Bengali	99.34	14.18	13.66	27.97

However, when we add the case when the genuine signature of the rest joins in to create pairs with the base signature of each user, the performance of our model decreases dramatically.

Table 13. *Performance on test set with random forgeries by other user genuine signatures*

Dataset	AUC (%)	ERR	FAR	FRR
SigComp2011	98	79.98	81.14	18.83
CEDAR	97.73	54.09	55.73	16.32
Bengali	99.32	30.9	30..96	27.97

4.3 Comparison

First, we create a simple Logistic Regression model which is used by the authors in [11] to compare with our classifier model. We draw the Precision-Recall curve for both classifiers on the test set from each dataset in ***Figure 21, 22*** and ***23***.

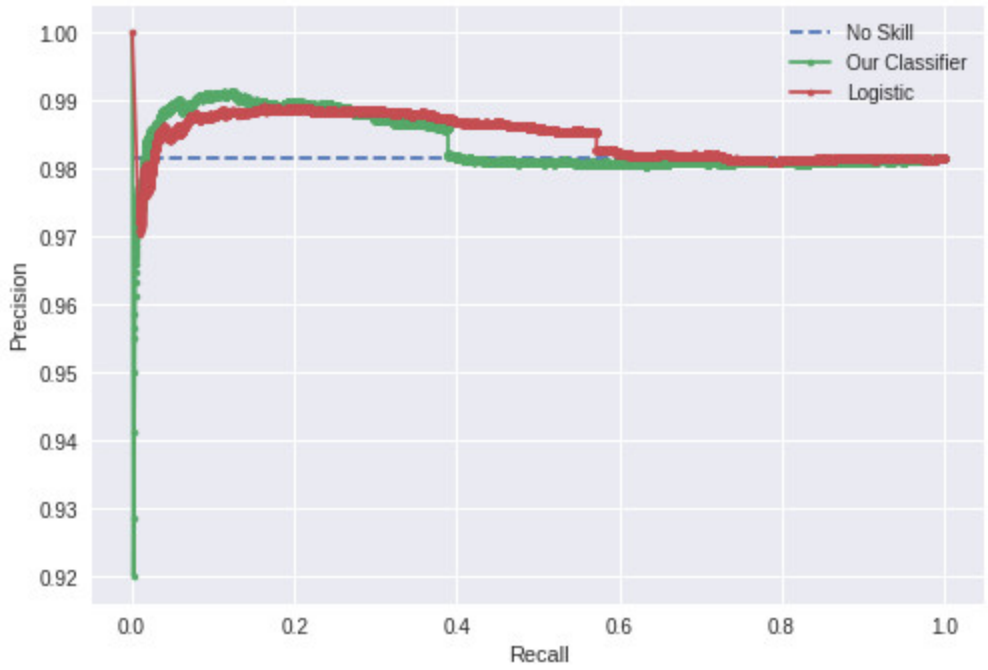

Figure 21. *PR Curve on SigComp2011 test set.*

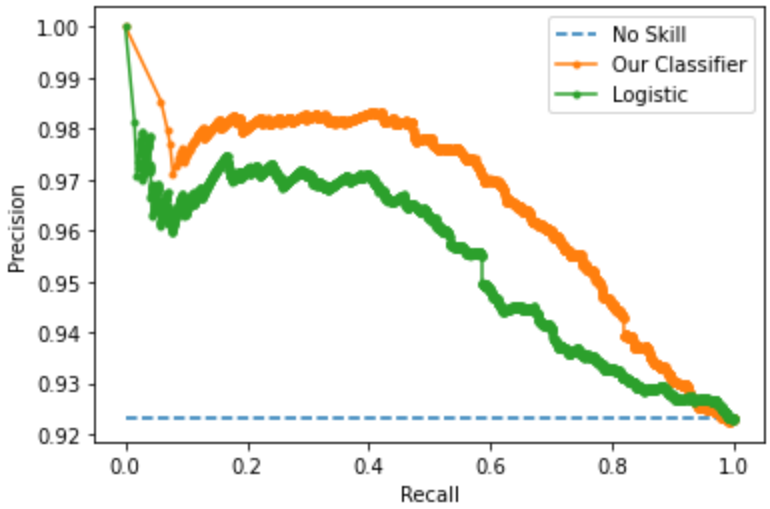

Figure 22. *PR Curve on CEDAR test set.*

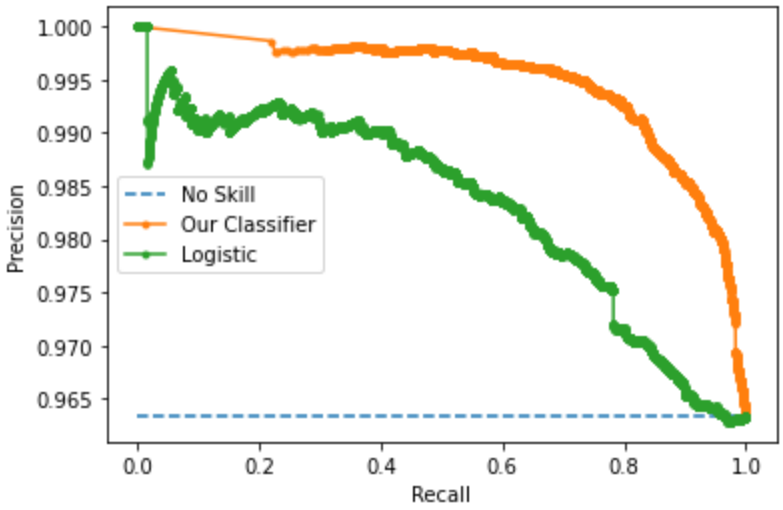

Figure 23. *PR Curve on BHSig260 test set.*

From the above PR Curves, we can see that our classifier always has higher AUC compared to the logistic regression, which means it is more efficient in our task when the positive class (forgery) is more important.

Moreover, the best result from ***Table 11, 12, 13*** can be used to compare with some state-of-the-art and the recent studies with Siamese Network in the following table:

Table 14. *State-of-the-art performance on CEDAR dataset (WD = Writer Dependent, WI = Writer Independent).*

Type	Features & algorithm	FRR	FAR	AER	ERR
WI [20]	Morphology (SVM)	12.39	11.23	11.81	11.59
WI [21]	Surroundness (NN)	-	8.33	8.33	8.33
WD [22]	Curvelet transform (OCSVM)	-	-	5.6	-
WD [23]	Chain code (SVM)	9.36	7.84	7.84	-
WD [18]	Feature learning (SVM)	-	-	-	4.63
WD [19]	Graph Matching	7.7	8.2	7.9	-
-	Our model	16	45	-	42.84

Table 15. *State-of-the-art performance on BHSig260 Dataset (WD = Writer Dependent, WI = Writer Independent).*

Language	Type	Features & algorithm	FRR	FAR
Bengali	WI [24]	SigNet	13.89	13.89
	WI [25]	Dutta et al	14.43	15.78
	WI [26]	Pal et al.	33.82	33.82
	-	**Our model**	27.97	**13.66**

5. Conclusion

From the result, it is so disappointing to say that our work is not efficient for various datasets. All the FAR, FRR, ERR on two easier dataset, CEDAR and SigComp, is too bad to be used. Somehow, we can achieve a good FAR result on the BHSig260 dataset which is slightly better than SigNet [24].

When we look at the AUC of PR Curve, our model is still stable in BHSig and two others. The increase of performance on the strategy of choosing which kind of random forgeries to feed on have shown that our model works well on detecting simple and skill forgeries, but may not fall behind when dealing with various kinds of random forgery signatures.

The difference between the accuracy (2) in the output of the Deep Triplet Ranking model can show that our selection method is not efficient enough. And the low accuracy and high FAR and FRR on CEDAR and SigComp dataset show that the strategy of processing from our dataset can not be the same for this task and need more improvement to work well.

With the above analysis, there are two things to summarize: firstly, our model has the ability to perform well on the dataset which is clean and not too small. Moreover , with the result slightly better than some studies in this field on the BHSig260, our method in preprocessing and pairing may be considered to be used in other studies.

Secondly, the bad results on CEDAR and SigComp show that the issue of our model is that it can not perform well on other datasets which have different characteristics compared to the train dataset. Our experiment on this structure also raises the question about the practicality of this Deep Triplet Ranking CNN architecture on other significant signature verification problems.

In the future, we need to find more suitable preprocessing methods for different kinds of datasets. We also need to focus on tuning hyperparameters of more powerful triplet ranking loss,

and may need to apply triplet online mining techniques to let the learning process of our model be better.

Despite the bad performance in two of three dataset, the results from the BHSig260 dataset make it possible to use our architecture on the problem which has the same dataset as the BHSig260 dataset.

References

1. Donato Impedovo and Giuseppe Pirlo. “Automatic Signature Verification: The State of the Art.” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *38(5):609–635*, September **2008**.

2. Abdul Salam Shah, M. N. A. Khan, and Asadullah Shah. “An Appraisal of Off-line Signature Verification Techniques.” *International Journal of Modern Education and Computer Science*, *4:67–75*, **2015**.

3. Hafemann, L.G.; Sabourin, R.; Oliveira, L.S. “Offline handwritten signature verification—Literature review.” *In Proceedings of the 2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA), Institute of Electrical and Electronics Engineers (IEEE)*, *Montreal, QC, Canada*, 28 November–1 December **2017**; *pp. 1–8*.

4. Diaz, M.; Ferrer, M.A.; Impedovo, D.; Malik, M.I.; Pirlo, G.; Plamondon, R. A. “Perspective analysis of handwritten signature technology.” *ACM Comput*. *Surv*. **2019**, *51, 1–39*, doi:10.1145/3274658.

5. Bouamra, W.; Djeddi, C.; Nini, B.; Diaz, M.; Siddiqi, I. “Towards the design of an offline signature verifier based on a small number of genuine samples for training.” *Expert Syst*. *Appl*. **2018**, *107, 182–195*, doi:10.1016/j.eswa.2018.04.035.

6. Hafemann, L.G.; Sabourin, R.; Oliveira, L.S. “Meta-Learning for fast classifier adaptation to new users of signature verification systems.” *IEEE Trans. Inf. Forensics Secur*. **2020**, *15, 1735–1745*, doi:10.1109/tifs.2019.2949425.

7. Hurieh Khalajzadeh, Mohammad Mansouri, and Mohammad Teshnehlab (**2012**). “Persian Signature Verification using Convolutional Neural Networks”. *In the International Journal of Engineering Research and Technology,* volume 1.

8. G. Koch, R. Zemel, R. Salakhutdinov. “Siamese neural networks for oneshot image recognition.” in: ICML, **2015**, pp. 1–8.

9. F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” ***2015*** *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815- 823, 2015.

10. Dey, Sounak & Dutta, Anjan & Toledo, J. & Ghosh, Suman & Lladós, Josep & Pal, Umapada. (**2017**). “SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification.”

11. Chhabra, O., & Chakraborty, S. (2019). “Siamese Triple Ranking Convolution Network in Signature Forgery Detection.“

12. Francois Chollet (**2016**). “Xception: Deep learning with Depthwise separable convolutions.” *arXiv preprint arXiv:1610.02357*

13. L. G. Hafemann, R. Sabourin, and L. S. Oliveira. “Learning features for offline handwritten signature verification using deep convolutional neural networks”. Pattern Recognition, 70:163–176, 2017. 2, 3

14. Muhammad Imran Malik, Marcus Liwicki, Linda Alewijnse, Wataru Ohyama, Michael Blumenstein, Charles E.H. Berger, and Bryan Found, “Signature Verification and Writer Identification Competitions for On- and Offline Skilled Forgeries (SigWiComp2013)”, in Proc. of the 12th Int. Conference on Document Analysis and Recognition (ICDAR), **2013**.

15. Marcus Liwicki, Elisa van den Heuvel, Bryan Found, Muhammad Imran Malik. "Forensic Signature Verification Competition 4NSigComp2010 – Detection of Simulated and Disguised Signatures", Proc. 12th Int. Conference on Frontiers in Handwriting Recognition, **2010**

16. Bromley, Jane, Bentz, James W, Bottou, Leon, Guyon, Isabelle, LeCun, Yann, Moore, Cliff, Sackinger, Eduard, and Shah, Roopak. “Signature verification using a siamese time delay neural network”. International Journal of Pattern Recognition and Artificial Intelligence, 7 (04):669–688, **1993**.

17. A. Rateria and S. Agarwal, "Off-line Signature Verification through Machine Learning", ***2018*** *5th IEEE Uttar Pradesh Section International Conference on Electrical Electronics and Computer Engineering UPCON 2018 8597090*.

18. skander, G.S., Sabourin, R., Granger, E.: Hybrid writer independent–writer dependent online signature verification system. IET biometrics 2(4), 169–181 (**2013**)

19. Khalajzadeh, H., Mansouri, M., Teshnehlab, M.: Persian signature verification using convolutional neural networks. International Journal of Engineering Research and Technology 1 (**2012**)

20. Kumar, R., Kundu, L., Chanda, B., Sharma, J.: A writer-independent o-line signature verification system based on signature morphology. In: Proceedings of the First International Conference on Intelligent Interactive Technologies and Multimedia. pp. 261–265. ACM (**2010**)

21. Kumar, R., Sharma, J., Chanda, B.: Writer-independent o-line signature verification using surroundedness feature. Pattern recognition letters 33(3), 301–308 (**2012**)

22. Nguyen, V., Kawazoe, Y., Wakabayashi, T., Pal, U., Blumenstein, M.: Performance analysis of the gradient feature and the modified direction feature for o-line signature verification. In: Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on. pp. 303–307. IEEE (**2010**)

23. Pal, S., Alaei, A., Pal, U., Blumenstein, M.: Performance of an o- line signature verification method based on texture features on a large indic-script signature dataset. In: Document Analysis Systems (DAS), 2016 12th IAPR Workshop on. pp. 72–77. IEEE (**2016**)

24. Batista, L., Granger, E., Sabourin, R.: Dynamic selection of generative–discriminative ensembles for o-line signature verification. Pattern Recognition 45(4), 1326–1340 (**2012**).

25. Ferrer, M.A., Alonso, J.B., Travieso, C.M.: Oine geometric parameters for automatic signature verification using fixed-point arithmetic. IEEE transactions on pattern analysis and machine intelligence 27(6), 993–997 (**2005**).

26. Hafemann, L.G., Sabourin, R., Oliveira, L.S.: Analyzing features learned for oine signature verification using deep cnns. In: Pattern Recognition (ICPR), 2016 23rd International Conference on. pp. 2989–2994. IEEE (**2016**)